



US011061613B2

(12) **United States Patent**
Baptist et al.

(10) **Patent No.:** **US 11,061,613 B2**
(45) **Date of Patent:** ***Jul. 13, 2021**

(54) **DYNAMIC RETENTION POLICIES AND
OPTIONAL DELETES**

(71) Applicant: **International Business Machines
Corporation**, Armonk, NY (US)

(72) Inventors: **Andrew D. Baptist**, Mt. Pleasant, WI
(US); **Bart R. Cilfone**, Marina del Rey,
CA (US); **Greg R. Dhuse**, Chicago, IL
(US); **Harsha Hegde**, Buffalo Grove,
IL (US); **Wesley B. Leggette**, Chicago,
IL (US); **Manish Motwani**, Chicago, IL
(US); **Jason K. Resch**, Chicago, IL
(US); **Ilya Volvovski**, Chicago, IL (US);
Ethan S. Wozniak, Park Ridge, IL
(US)

(73) Assignee: **INTERNATIONAL BUSINESS
MACHINES CORPORATION**,
Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 69 days.

This patent is subject to a terminal dis-
claimer.

(21) Appl. No.: **16/520,511**

(22) Filed: **Jul. 24, 2019**

(65) **Prior Publication Data**

US 2019/0347046 A1 Nov. 14, 2019

Related U.S. Application Data

(63) Continuation of application No. 15/485,095, filed on
Apr. 11, 2017, now Pat. No. 10,545,699.

(51) **Int. Cl.**

G06F 3/06 (2006.01)

G06F 16/11 (2019.01)

(Continued)

(52) **U.S. Cl.**

CPC **G06F 3/0659** (2013.01); **G06F 3/0604**
(2013.01); **G06F 3/064** (2013.01);

(Continued)

(58) **Field of Classification Search**

CPC **G06F 3/0659**; **G06F 3/0604**; **G06F 3/0619**;
G06F 3/067; **G06F 3/064**; **G06F 3/0649**;

(Continued)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,092,732 A 5/1978 Ouchi

5,454,101 A 9/1995 Mackay et al.

(Continued)

OTHER PUBLICATIONS

Chung; An Automatic Data Segmentation Method for 3D Measured
Data Points; National Taiwan University; pp. 1-8; 1998.

(Continued)

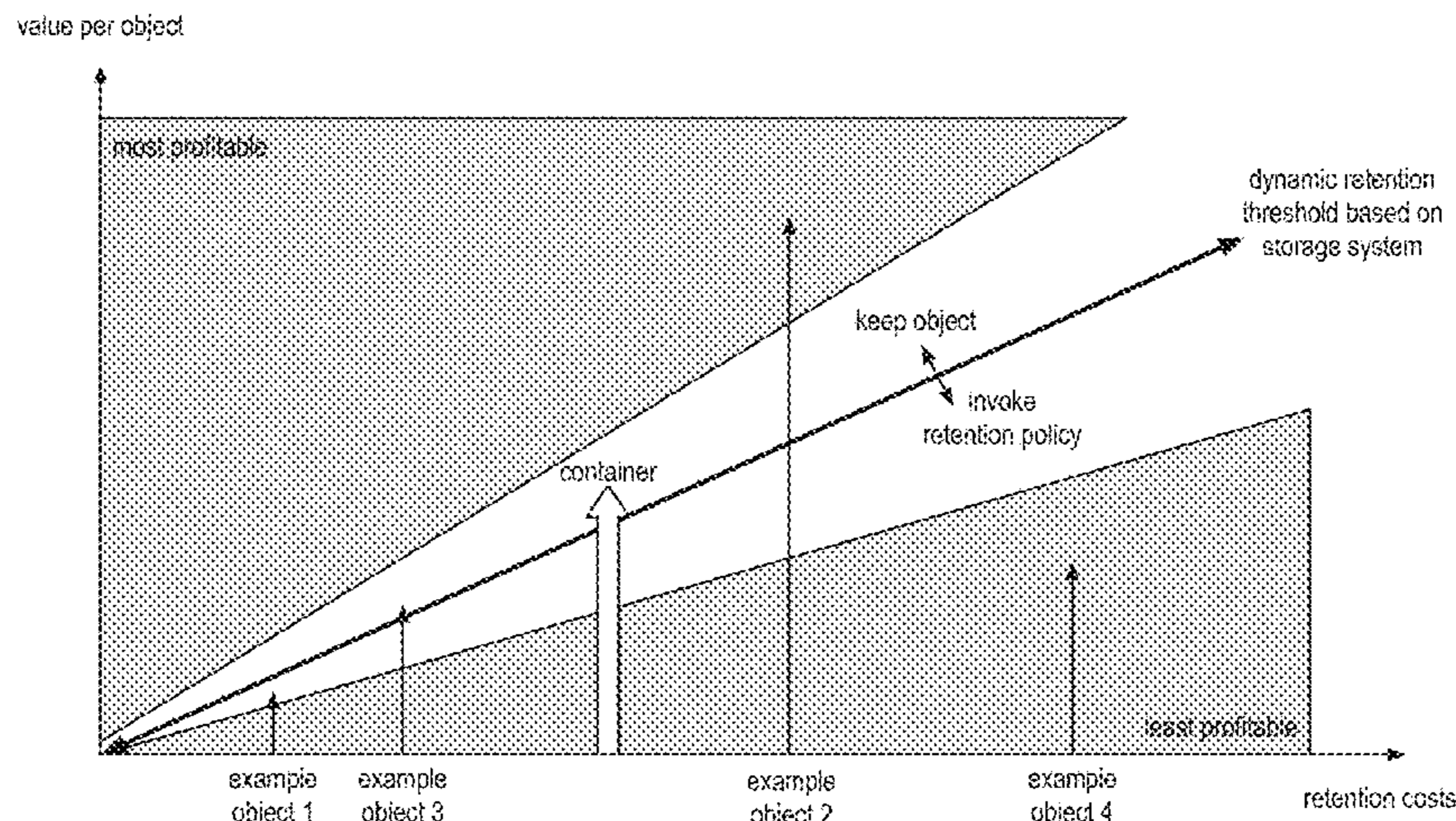
Primary Examiner — James E Richardson

(74) *Attorney, Agent, or Firm* — Anthony V. England;
Andrew D. Wright; Roberts Calderon Safran & Cole, P.C.

(57) **ABSTRACT**

A computer readable storage device includes a first memory
section that stores operational instructions that, when
executed, cause a computing device to, as data accesses
occur for a plurality of data objects of a storage container,
update object values to produce updated object values,
update object retention costs to produce updated object
retention costs, adjust a dynamic retention threshold based
on the updated object values and the updated object reten-
tion costs and update a data object retention policy for a data
object based on the dynamic retention threshold to produce
an updated retention policy for the data object. The com-
puter readable storage device includes a second memory
section that stores operational instructions that, when
executed, cause the computing device to, when a data access
is a deletion request, utilizing a current updated data object

(Continued)



(56)

References Cited

OTHER PUBLICATIONS

Shamir; How to Share a Secret; Communications of the ACM; vol. 22, No. 11; Nov. 1979; pp. 612-613.

Smith; Lightweight Directory Access Protocol (LDAP): String Representation of Search Filters; IETF Network Working Group; RFC 4515; Jun. 2006; pp. 1-12.

Smith; Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator; IETF Network Working Group; RFC 4516; Jun. 2006; pp. 1-15.

Wildi; Java iSCSI Initiator; Master Thesis; Department of Computer and Information Science, University of Konstanz; Feb. 2007; 60 pgs.

Xin, et al.; Evaluation of Distributed Recovery in Large-Scale Storage Systems; 13th IEEE International Symposium on High Performance Distributed Computing; Jun. 2004; pp. 172-181.

Zeilenga; Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map; IETF Network Working Group; RFC 4510; Jun. 2006; pp. 1-8.

Zeilenga; Lightweight Directory Access Protocol (LDAP): Directory Information Models; IETF Network Working Group; RFC 4512; Jun. 2006; pp. 1-49.

Zeilenga; Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation; IETF Network Working Group; RFC 4518; Jun. 2006; pp. 1-14.

Zeilenga; Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names; IETF Network Working Group; RFC 4514; Jun. 2006; pp. 1-15.

List of IBM Patents or Patent Applications Treated as Related, dated Mar. 2, 2021, 1 page.

* cited by examiner

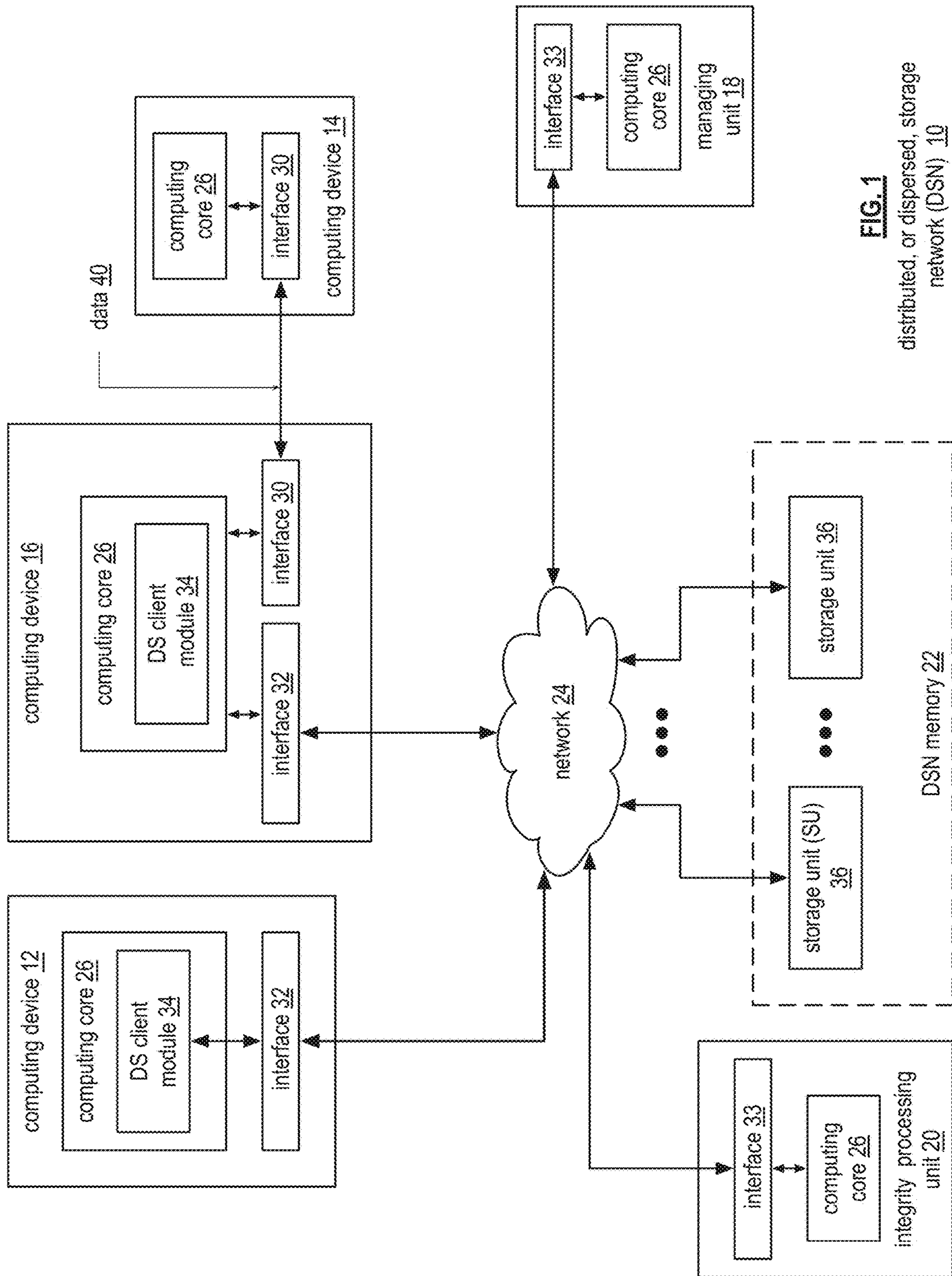


FIG. 1

distributed, or dispersed, storage network (DSN) 10

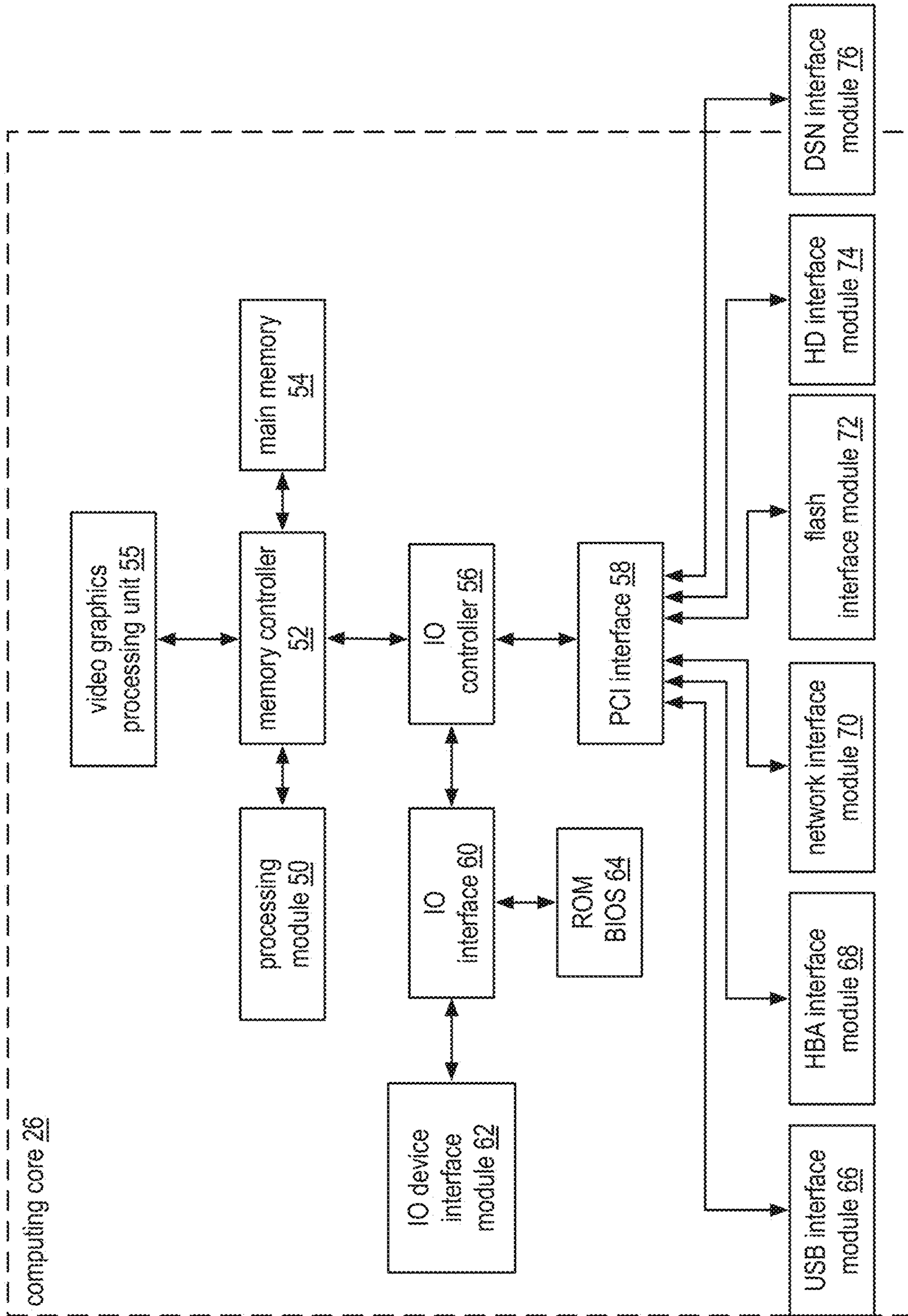


FIG. 2

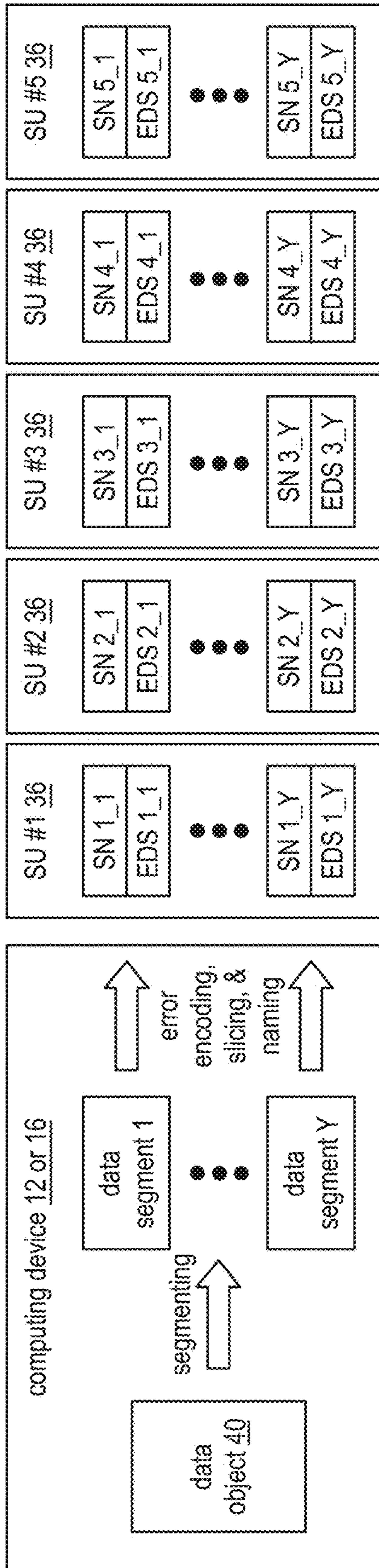


FIG. 3

SN = slice name
EDS = encoded data slice

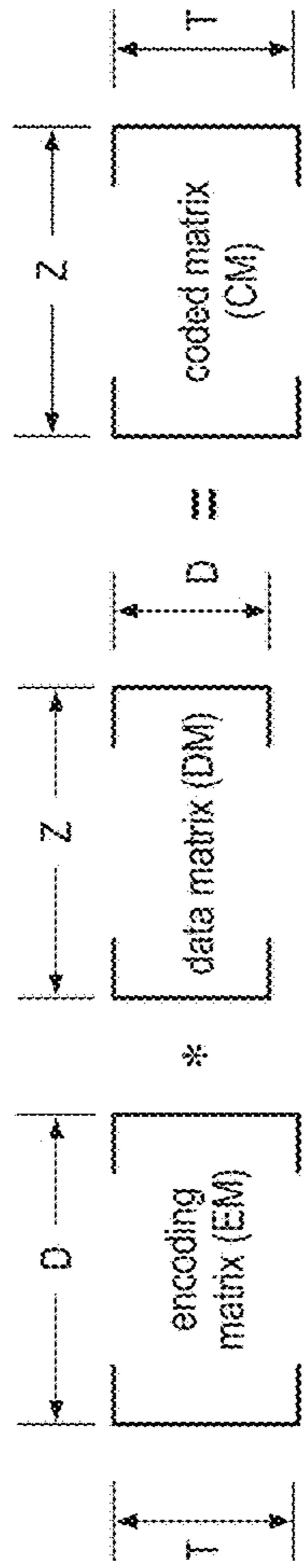


FIG. 4

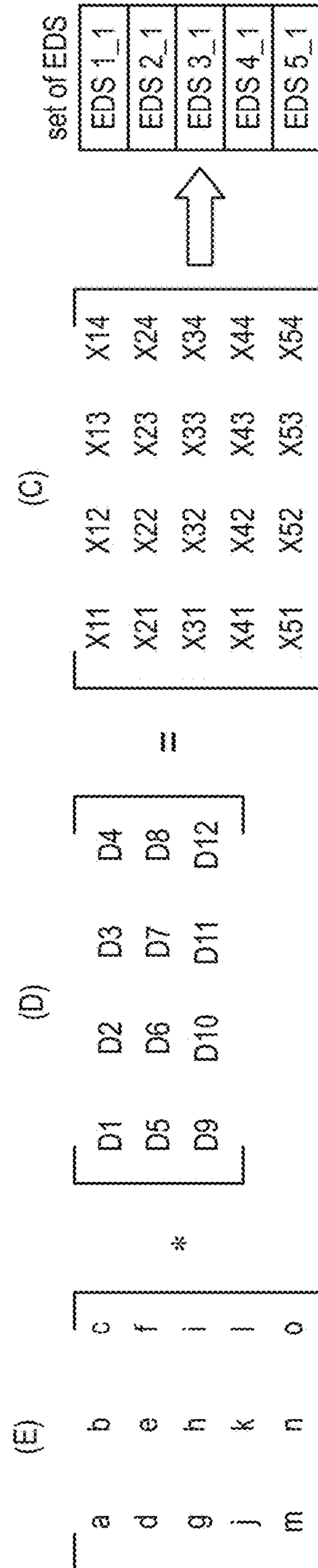


FIG. 5

slice name 80			
pillar #	data segment #	vault ID	data object ID
			rev. info

FIG. 6

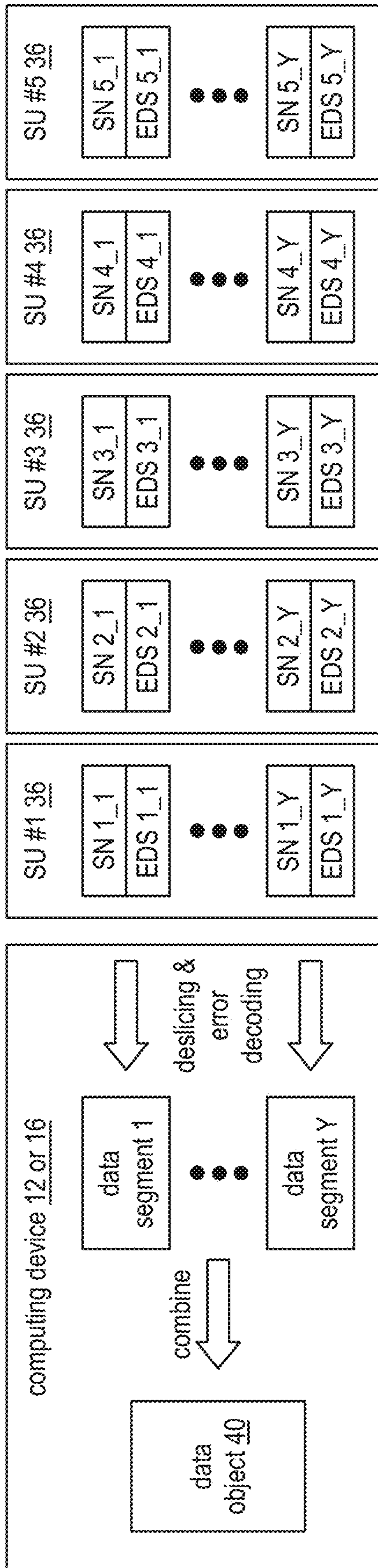


FIG. 7

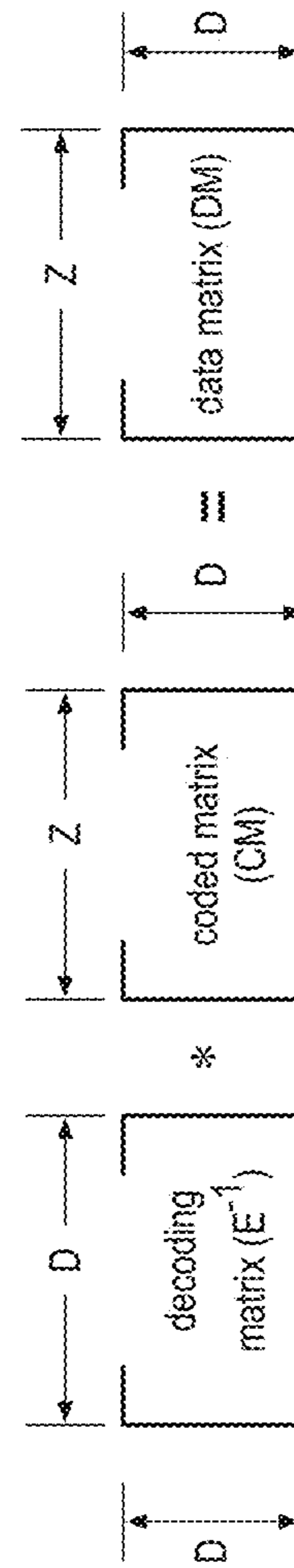


FIG. 8

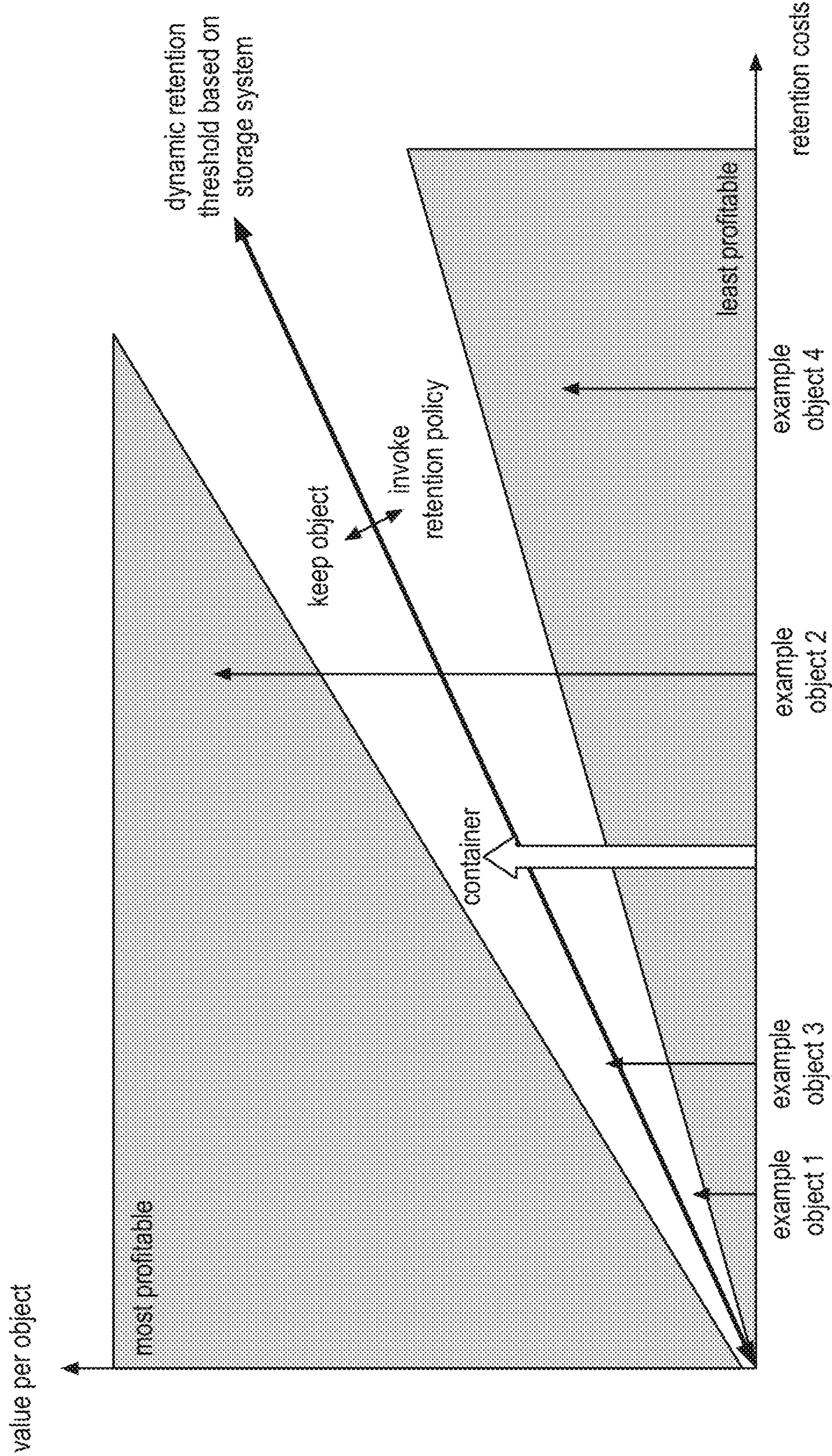


FIG. 9

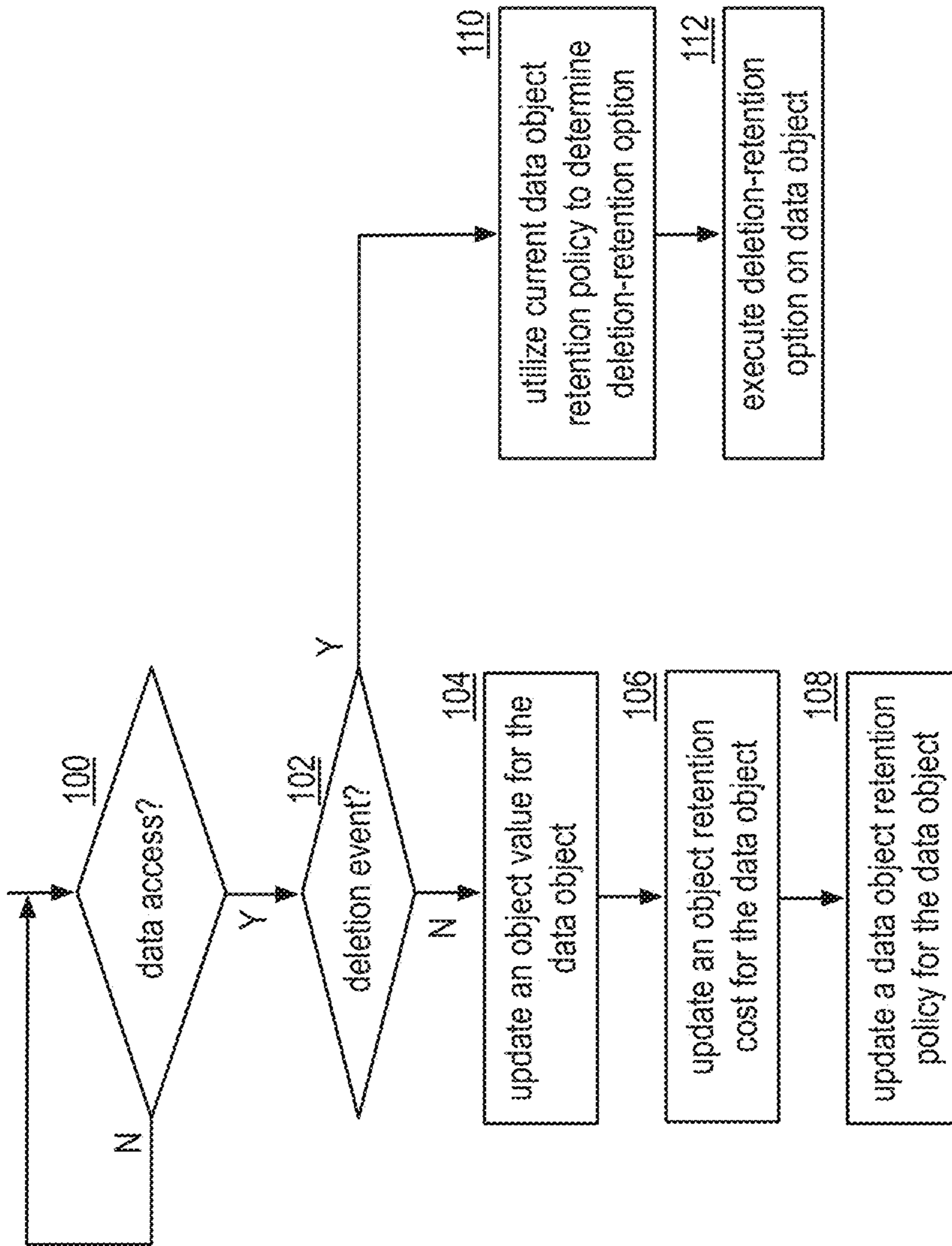


FIG. 10

1**DYNAMIC RETENTION POLICIES AND
OPTIONAL DELETES**STATEMENT REGARDING FEDERALLY
SPONSORED RESEARCH OR DEVELOPMENT

Not applicable.

INCORPORATION-BY-REFERENCE OF
MATERIAL SUBMITTED ON A COMPACT
DISC

Not applicable.

BACKGROUND OF THE INVENTION

Technical Field of the Invention

This invention relates generally to computer networks and more particularly to dispersing error encoded data.

Description of Related Art

Computing devices are known to communicate data, process data, and/or store data. Such computing devices range from wireless smart phones, laptops, tablets, personal computers (PC), work stations, and video game devices, to data centers that support millions of web searches, stock trades, or on-line purchases every day. In general, a computing device includes a central processing unit (CPU), a memory system, user input/output interfaces, peripheral device interfaces, and an interconnecting bus structure.

As is further known, a computer may effectively extend its CPU by using "cloud computing" to perform one or more computing functions (e.g., a service, an application, an algorithm, an arithmetic logic function, etc.) on behalf of the computer. Further, for large services, applications, and/or functions, cloud computing may be performed by multiple cloud computing resources in a distributed manner to improve the response time for completion of the service, application, and/or function. For example, Hadoop is an open source software framework that supports distributed applications enabling application execution by thousands of computers.

In addition to cloud computing, a computer may use "cloud storage" as part of its memory system. As is known, cloud storage enables a user, via its computer, to store files, applications, etc. on an Internet storage system. The Internet storage system may include a RAID (redundant array of independent disks) system and/or a dispersed storage system that uses an error correction scheme to encode data for storage.

BRIEF DESCRIPTION OF THE SEVERAL
VIEWS OF THE DRAWING(S)

FIG. 1 is a schematic block diagram of an embodiment of a dispersed or distributed storage network (DSN) in accordance with the present invention;

FIG. 2 is a schematic block diagram of an embodiment of a computing core in accordance with the present invention;

FIG. 3 is a schematic block diagram of an example of dispersed storage error encoding of data in accordance with the present invention;

FIG. 4 is a schematic block diagram of a generic example of an error encoding function in accordance with the present invention;

2

FIG. 5 is a schematic block diagram of a specific example of an error encoding function in accordance with the present invention;

FIG. 6 is a schematic block diagram of an example of a slice name of an encoded data slice (EDS) in accordance with the present invention;

FIG. 7 is a schematic block diagram of an example of dispersed storage error decoding of data in accordance with the present invention;

FIG. 8 is a schematic block diagram of a generic example of an error decoding function in accordance with the present invention;

FIG. 9 is a graph diagram of an embodiment of a dynamic retention policy in accordance with the present invention; and

FIG. 10 is a logic diagram of an example of a method of a dynamic retention policy in accordance with the present invention.

DETAILED DESCRIPTION OF THE
INVENTION

FIG. 1 is a schematic block diagram of an embodiment of a dispersed, or distributed, storage network (DSN) 10 that includes a plurality of computing devices 12-16, a managing unit 18, an integrity processing unit 20, and a DSN memory 22. The components of the DSN 10 are coupled to a network 24, which may include one or more wireless and/or wire lined communication systems; one or more non-public intranet systems and/or public internet systems; and/or one or more local area networks (LAN) and/or wide area networks (WAN).

The DSN memory 22 includes a plurality of storage units 36 that may be located at geographically different sites (e.g., one in Chicago, one in Milwaukee, etc.), at a common site, or a combination thereof. For example, if the DSN memory 22 includes eight storage units 36, each storage unit is located at a different site. As another example, if the DSN memory 22 includes eight storage units 36, all eight storage units are located at the same site. As yet another example, if the DSN memory 22 includes eight storage units 36, a first pair of storage units are at a first common site, a second pair of storage units are at a second common site, a third pair of storage units are at a third common site, and a fourth pair of storage units are at a fourth common site. Note that a DSN memory 22 may include more or less than eight storage units 36. Further note that each storage unit 36 includes a computing core (as shown in FIG. 2, or components thereof) and a plurality of memory devices for storing dispersed error encoded data.

Each of the computing devices 12-16, the managing unit 18, and the integrity processing unit 20 include a computing core 26, which includes network interfaces 30-33. Computing devices 12-16 may each be a portable computing device and/or a fixed computing device. A portable computing device may be a social networking device, a gaming device, a cell phone, a smart phone, a digital assistant, a digital music player, a digital video player, a laptop computer, a handheld computer, a tablet, a video game controller, and/or any other portable device that includes a computing core. A fixed computing device may be a computer (PC), a computer server, a cable set-top box, a satellite receiver, a television set, a printer, a fax machine, home entertainment equipment, a video game console, and/or any type of home or office computing equipment. Note that each of the managing unit 18 and the integrity processing unit 20 may be separate computing devices, may be a common computing device,

and/or may be integrated into one or more of the computing devices **12-16** and/or into one or more of the storage units **36**.

Each interface **30**, **32**, and **33** includes software and hardware to support one or more communication links via the network **24** indirectly and/or directly. For example, interface **30** supports a communication link (e.g., wired, wireless, direct, via a LAN, via the network **24**, etc.) between computing devices **14** and **16**. As another example, interface **32** supports communication links (e.g., a wired connection, a wireless connection, a LAN connection, and/or any other type of connection to/from the network **24**) between computing devices **12** and **16** and the DSN memory **22**. As yet another example, interface **33** supports a communication link for each of the managing unit **18** and the integrity processing unit **20** to the network **24**.

Computing devices **12** and **16** include a dispersed storage (DS) client module **34**, which enables the computing device to dispersed storage error encode and decode data (e.g., data **40**) as subsequently described with reference to one or more of FIGS. **3-8**. In this example embodiment, computing device **16** functions as a dispersed storage processing agent for computing device **14**. In this role, computing device **16** dispersed storage error encodes and decodes data on behalf of computing device **14**. With the use of dispersed storage error encoding and decoding, the DSN **10** is tolerant of a significant number of storage unit failures (the number of failures is based on parameters of the dispersed storage error encoding function) without loss of data and without the need for a redundant or backup copies of the data. Further, the DSN **10** stores data for an indefinite period of time without data loss and in a secure manner (e.g., the system is very resistant to unauthorized attempts at accessing the data).

In operation, the managing unit **18** performs DS management services. For example, the managing unit **18** establishes distributed data storage parameters (e.g., vault creation, distributed storage parameters, security parameters, billing information, user profile information, etc.) for computing devices **12-14** individually or as part of a group of user devices. As a specific example, the managing unit **18** coordinates creation of a vault (e.g., a virtual memory block associated with a portion of an overall namespace of the DSN) within the DSN memory **22** for a user device, a group of devices, or for public access and establishes per vault dispersed storage (DS) error encoding parameters for a vault. The managing unit **18** facilitates storage of DS error encoding parameters for each vault by updating registry information of the DSN **10**, where the registry information may be stored in the DSN memory **22**, a computing device **12-16**, the managing unit **18**, and/or the integrity processing unit **20**.

The managing unit **18** creates and stores user profile information (e.g., an access control list (ACL)) in local memory and/or within memory of the DSN memory **22**. The user profile information includes authentication information, permissions, and/or the security parameters. The security parameters may include encryption/decryption scheme, one or more encryption keys, key generation scheme, and/or data encoding/decoding scheme.

The managing unit **18** creates billing information for a particular user, a user group, a vault access, public vault access, etc. For instance, the managing unit **18** tracks the number of times a user accesses a non-public vault and/or public vaults, which can be used to generate a per-access billing information. In another instance, the managing unit **18** tracks the amount of data stored and/or retrieved by a user

device and/or a user group, which can be used to generate a per-data-amount billing information.

As another example, the managing unit **18** performs network operations, network administration, and/or network maintenance. Network operations includes authenticating user data allocation requests (e.g., read and/or write requests), managing creation of vaults, establishing authentication credentials for user devices, adding/deleting components (e.g., user devices, storage units, and/or computing devices with a DS client module **34**) to/from the DSN **10**, and/or establishing authentication credentials for the storage units **36**. Network administration includes monitoring devices and/or units for failures, maintaining vault information, determining device and/or unit activation status, determining device and/or unit loading, and/or determining any other system level operation that affects the performance level of the DSN **10**. Network maintenance includes facilitating replacing, upgrading, repairing, and/or expanding a device and/or unit of the DSN **10**.

The integrity processing unit **20** performs rebuilding of 'bad' or missing encoded data slices. At a high level, the integrity processing unit **20** performs rebuilding by periodically attempting to retrieve/list encoded data slices, and/or slice names of the encoded data slices, from the DSN memory **22**. For retrieved encoded slices, they are checked for errors due to data corruption, outdated version, etc. If a slice includes an error, it is flagged as a 'bad' slice. For encoded data slices that were not received and/or not listed, they are flagged as missing slices. Bad and/or missing slices are subsequently rebuilt using other retrieved encoded data slices that are deemed to be good slices to produce rebuilt slices. The rebuilt slices are stored in the DSN memory **22**.

FIG. **2** is a schematic block diagram of an embodiment of a computing core **26** that includes a processing module **50**, a memory controller **52**, main memory **54**, a video graphics processing unit **55**, an input/output (IO) controller **56**, a peripheral component interconnect (PCI) interface **58**, an IO interface module **60**, at least one IO device interface module **62**, a read only memory (ROM) basic input output system (BIOS) **64**, and one or more memory interface modules. The one or more memory interface module(s) includes one or more of a universal serial bus (USB) interface module **66**, a host bus adapter (HBA) interface module **68**, a network interface module **70**, a flash interface module **72**, a hard drive interface module **74**, and a DSN interface module **76**.

The DSN interface module **76** functions to mimic a conventional operating system (OS) file system interface (e.g., network file system (NFS), flash file system (FFS), disk file system (DFS), file transfer protocol (FTP), web-based distributed authoring and versioning (WebDAV), etc.) and/or a block memory interface (e.g., small computer system interface (SCSI), internet small computer system interface (iSCSI), etc.). The DSN interface module **76** and/or the network interface module **70** may function as one or more of the interface **30-33** of FIG. **1**. Note that the IO device interface module **62** and/or the memory interface modules **66-76** may be collectively or individually referred to as IO ports.

FIG. **3** is a schematic block diagram of an example of dispersed storage error encoding of data. When a computing device **12** or **16** has data to store it disperse storage error encodes the data in accordance with a dispersed storage error encoding process based on dispersed storage error encoding parameters. The dispersed storage error encoding parameters include an encoding function (e.g., information dispersal algorithm, Reed-Solomon, Cauchy Reed-Solomon, systematic encoding, non-systematic encoding, on-line

5

codes, etc.), a data segmenting protocol (e.g., data segment size, fixed, variable, etc.), and per data segment encoding values. The per data segment encoding values include a total, or pillar width, number (T) of encoded data slices per encoding of a data segment (i.e., in a set of encoded data slices); a decode threshold number (D) of encoded data slices of a set of encoded data slices that are needed to recover the data segment; a read threshold number (R) of encoded data slices to indicate a number of encoded data slices per set to be read from storage for decoding of the data segment; and/or a write threshold number (W) to indicate a number of encoded data slices per set that must be accurately stored before the encoded data segment is deemed to have been properly stored. The dispersed storage error encoding parameters may further include slicing information (e.g., the number of encoded data slices that will be created for each data segment) and/or slice security information (e.g., per encoded data slice encryption, compression, integrity checksum, etc.).

In the present example, Cauchy Reed-Solomon has been selected as the encoding function (a generic example is shown in FIG. 4 and a specific example is shown in FIG. 5); the data segmenting protocol is to divide the data object into fixed sized data segments; and the per data segment encoding values include: a pillar width of 5, a decode threshold of 3, a read threshold of 4, and a write threshold of 4. In accordance with the data segmenting protocol, the computing device 12 or 16 divides the data (e.g., a file (e.g., text, video, audio, etc.), a data object, or other data arrangement) into a plurality of fixed sized data segments (e.g., 1 through Y of a fixed size in range of Kilo-bytes to Tera-bytes or more). The number of data segments created is dependent of the size of the data and the data segmenting protocol.

The computing device 12 or 16 then disperse storage error encodes a data segment using the selected encoding function (e.g., Cauchy Reed-Solomon) to produce a set of encoded data slices. FIG. 4 illustrates a generic Cauchy Reed-Solomon encoding function, which includes an encoding matrix (EM), a data matrix (DM), and a coded matrix (CM). The size of the encoding matrix (EM) is dependent on the pillar width number (T) and the decode threshold number (D) of selected per data segment encoding values. To produce the data matrix (DM), the data segment is divided into a plurality of data blocks and the data blocks are arranged into D number of rows with Z data blocks per row. Note that Z is a function of the number of data blocks created from the data segment and the decode threshold number (D). The coded matrix is produced by matrix multiplying the data matrix by the encoding matrix.

FIG. 5 illustrates a specific example of Cauchy Reed-Solomon encoding with a pillar number (T) of five and decode threshold number of three. In this example, a first data segment is divided into twelve data blocks (D1-D12). The coded matrix includes five rows of coded data blocks, where the first row of X11-X14 corresponds to a first encoded data slice (EDS 1_1), the second row of X21-X24 corresponds to a second encoded data slice (EDS 2_1), the third row of X31-X34 corresponds to a third encoded data slice (EDS 3_1), the fourth row of X41-X44 corresponds to a fourth encoded data slice (EDS 4_1), and the fifth row of X51-X54 corresponds to a fifth encoded data slice (EDS 5_1). Note that the second number of the EDS designation corresponds to the data segment number.

Returning to the discussion of FIG. 3, the computing device also creates a slice name (SN) for each encoded data slice (EDS) in the set of encoded data slices. A typical format for a slice name 80 is shown in FIG. 6. As shown, the slice

6

name (SN) 80 includes a pillar number of the encoded data slice (e.g., one of 1-T), a data segment number (e.g., one of 1-Y), a vault identifier (ID), a data object identifier (ID), and may further include revision level information of the encoded data slices. The slice name functions as, at least part of, a DSN address for the encoded data slice for storage and retrieval from the DSN memory 22.

As a result of encoding, the computing device 12 or 16 produces a plurality of sets of encoded data slices, which are provided with their respective slice names to the storage units for storage. As shown, the first set of encoded data slices includes EDS 1_1 through EDS 5_1 and the first set of slice names includes SN 1_1 through SN 5_1 and the last set of encoded data slices includes EDS 1_Y through EDS 5_Y and the last set of slice names includes SN 1_Y through SN 5_Y.

FIG. 7 is a schematic block diagram of an example of dispersed storage error decoding of a data object that was dispersed storage error encoded and stored in the example of FIG. 4. In this example, the computing device 12 or 16 retrieves from the storage units at least the decode threshold number of encoded data slices per data segment. As a specific example, the computing device retrieves a read threshold number of encoded data slices.

To recover a data segment from a decode threshold number of encoded data slices, the computing device uses a decoding function as shown in FIG. 8. As shown, the decoding function is essentially an inverse of the encoding function of FIG. 4. The coded matrix includes a decode threshold number of rows (e.g., three in this example) and the decoding matrix is an inversion of the encoding matrix that includes the corresponding rows of the coded matrix. For example, if the coded matrix includes rows 1, 2, and 4, the encoding matrix is reduced to rows 1, 2, and 4, and then inverted to produce the decoding matrix.

FIG. 9 is a graph diagram of an embodiment of a dynamic retention policy. As illustrated in the graph, the x-axis represents retention costs and the y-axis represents object values. A retention cost for an object is calculated based on one or more of a cost of access per object, a cost per read, a cost per input-output process (TOP), storage costs, lower quality of service (QoS) and a user subscription. An object value is calculated based on one or more of a frequency of access, an object type, a cost of access, company regulations, government regulations and overwrite occurrences.

The retention policy for a data object or container may be updated on every access request, updated every certain number of access requests (e.g., every 50), updated according to a time schedule (e.g., once a minute, once an hour, etc.), or updated according to a user request. As an example, for every non-delete access request for a data object, a computing device applies inputs based on the request to a heuristic function, performs the heuristic function, updates a retention policy based on the results of the heuristic function, and stores the retention policy within an object store along with the data object and container metadata. Further, for every delete request, the computing device may check the retention policy to determine if the request is optional from an end user perspective. The retention policy may be checked by reading one or more of the object and container metadata.

The weighted line is a function that represents a dynamic retention threshold based on the storage system. The dynamic retention threshold is calculated based on overall storage factors of the storage system which include one or more of memory availability, memory utilization, reliability, data throughput rate and retention cost-value object com-

parisons (e.g., object 4 higher probability of delete than object 1). For example, when a system has a desired level of profitability, the threshold may be adjusted (e.g., shifted up the y-axis, function changed (e.g., to an exponential function), etc.) so that the system tends to keep more profitable objects in order to obtain the desired level of profitability. In one example, the system determines to keep objects that are above the dynamic retention threshold and invokes a retention policy for objects that are below the dynamic retention threshold.

As a specific example, a computing device determines object 1 and object 4 are below the dynamic retention threshold and determines objects 2 and 3 are above the dynamic retention threshold. Having determined that data objects 1 and 4 are below the dynamic retention threshold, the computing device invokes the retention policy for objects 1 and 4. The computing device further determines that a container value is above the dynamic retention threshold and thus determines not to invoke the retention policy for the container (e.g., keeps the container). Having determined to invoke the retention policy for objects 1 and 4, the computing device determines each data object's retention policy. For example, the computing device determines that object 1's retention policy is deletion and data object scrub without archiving and object 4's retention policy is deletion with archiving. Having determined the retention policies for objects 1 and 4, the computing device then executes the retention policies on objects 1 and 4. For example, the computing device deletes objects 1 and 4 from the container, scrubs object 1 (e.g., overwriting data object memory section with 0's one or more times, trimming the blocks of data corresponding to the data object, etc.), and archives object 4.

Other retention policies that may be invoked include a deletion with archiving option, a deletion without archiving option, a deletion and data object scrub without archiving option, a deletion prohibition option (e.g., government data, security policy, etc.), a deletion restriction option based on one or more user parameters (e.g., user status, type of data, paid for service), and a subsequent data disposal option. Note that each object's retention policy may be updated to include a different one of the data object retention policies.

FIG. 10 is a logic diagram of an example of a method of a dynamic retention policy in accordance with the present invention. The method begins at step 100, where a computing device of a dispersed storage network (DSN) determines if a data access has occurred for a data object of a storage container within the DSN. When the data access has not occurred, the method loops back to step 100. When the data access has occurred, the method continues to step 102, where the computing device determines if the data access is a deletion event.

When the data access is not a deletion event, the method continues to step 104, where the computing device updates, for each of the data accesses, an object value for the data object to produce an updated object value. As an example, the computing device determines an updated value factor based on one or more of frequency of the data accesses for the data object, a type of the data object, a cost associated with one or more of the data access, a regulation regarding deletion-retention of the data object, and overwriting occurrences (e.g., editing, revisions, etc.) of the data object. The computing device then changes a then current object value based on the updated value factor to produce the updated object value. As another example, the computing device updates, for each of the data accesses, a storage container

value to produce an updated storage container value (e.g., compilation of storage values for the data objects in the container).

The method continues to step 106, where the computing device updates, for each of the data accesses, an object retention cost for the data object to produce an updated object retention cost. As an example, the computing device determines an updated cost factor based on one or more of cost per access of the data object, cost per read of the data object, cost per input-output processing (TOP) of the data object, storage costs of the data object and user subscription criteria regarding the data object (e.g., quality of service (QoS), level of storage service, reliability (e.g., wider pillar number), etc.). Having produced the updated cost factor, the computing device changes a then current object retention cost based on the updated cost factor to produce the updated retention cost. As another example, the computing device updates, for each of the data accesses, a storage container cost to produce an updated storage container cost (e.g., compilation of storage costs for the data objects in the container).

The method continues to step 108, where the computing device updates, for each of the data accesses, a data object retention policy for the data object based on the updated object value and the updated object retentions costs. The data object retention policy includes one or more of a deletion with archiving option, a deletion without archiving option, a deletion and data object scrub without archiving option, a deletion prohibition option, a deletion restriction option based on one or more user parameters (e.g., user status, type of data, paid for service, etc.) and a subsequent data disposal option.

As an example of updating the data object retention policy, the computing device executes a heuristic function using the updated object value and the updated object retention costs as inputs. Having executed the heuristic function, the computing device then updates a then current data object retention policy based on a result of the heuristic function to produce the updated data object retention policy. As another example, the computing device updates, for each of the data accesses, the data object retention policy for the data object based on the updated object value, the updated object retentions costs, the updated storage container value, and the updated storage container cost.

When the data access is a deletion event, the method continues to step 110, where the computing device utilizes a current updated data object retention policy to determine a deletion-retention option for the data object. As an example of the deletion event, the computing device receives a deletion request from a user device associated with the data object to produce the deletion event. As another example of the deletion event, the deletion event is triggered by the DSN when a container, a data object or multiple data objects falls below a dynamic retention threshold. For example, the computing device determines a retention threshold for the data object based on the updated data object retention policy, and when value-cost of the data object compares unfavorably to the retention threshold, initiates the deletion event.

The method continues to step 112, where the computing device executes the deletion-retention option on the data object. As an example, the computing device receives a deletion request for a plurality of data objects stored in the storage container, wherein the plurality of data objects includes the data object. The computing device then, for each data object of the plurality of data objects, determines a then current corresponding data object retention policy.

Having determined the then current corresponding data object retention policy, the computing device executes the then current corresponding data object retention policy.

It is noted that terminologies as may be used herein such as bit stream, stream, signal sequence, etc. (or their equivalents) have been used interchangeably to describe digital information whose content corresponds to any of a number of desired types (e.g., data, video, speech, audio, etc. any of which may generally be referred to as 'data').

As may be used herein, the terms "substantially" and "approximately" provides an industry-accepted tolerance for its corresponding term and/or relativity between items. Such an industry-accepted tolerance ranges from less than one percent to fifty percent and corresponds to, but is not limited to, component values, integrated circuit process variations, temperature variations, rise and fall times, and/or thermal noise. Such relativity between items ranges from a difference of a few percent to magnitude differences. As may also be used herein, the term(s) "configured to", "operably coupled to", "coupled to", and/or "coupling" includes direct coupling between items and/or indirect coupling between items via an intervening item (e.g., an item includes, but is not limited to, a component, an element, a circuit, and/or a module) where, for an example of indirect coupling, the intervening item does not modify the information of a signal but may adjust its current level, voltage level, and/or power level. As may further be used herein, inferred coupling (i.e., where one element is coupled to another element by inference) includes direct and indirect coupling between two items in the same manner as "coupled to". As may even further be used herein, the term "configured to", "operable to", "coupled to", or "operably coupled to" indicates that an item includes one or more of power connections, input(s), output(s), etc., to perform, when activated, one or more its corresponding functions and may further include inferred coupling to one or more other items. As may still further be used herein, the term "associated with", includes direct and/or indirect coupling of separate items and/or one item being embedded within another item.

As may be used herein, the term "compares favorably", indicates that a comparison between two or more items, signals, etc., provides a desired relationship. For example, when the desired relationship is that signal 1 has a greater magnitude than signal 2, a favorable comparison may be achieved when the magnitude of signal 1 is greater than that of signal 2 or when the magnitude of signal 2 is less than that of signal 1. As may be used herein, the term "compares unfavorably", indicates that a comparison between two or more items, signals, etc., fails to provide the desired relationship.

As may also be used herein, the terms "processing module", "processing circuit", "processor", and/or "processing unit" may be a single processing device or a plurality of processing devices. Such a processing device may be a microprocessor, micro-controller, digital signal processor, microcomputer, central processing unit, field programmable gate array, programmable logic device, state machine, logic circuitry, analog circuitry, digital circuitry, and/or any device that manipulates signals (analog and/or digital) based on hard coding of the circuitry and/or operational instructions. The processing module, module, processing circuit, and/or processing unit may be, or further include, memory and/or an integrated memory element, which may be a single memory device, a plurality of memory devices, and/or embedded circuitry of another processing module, module, processing circuit, and/or processing unit. Such a memory device may be a read-only memory, random access memory,

volatile memory, non-volatile memory, static memory, dynamic memory, flash memory, cache memory, and/or any device that stores digital information. Note that if the processing module, module, processing circuit, and/or processing unit includes more than one processing device, the processing devices may be centrally located (e.g., directly coupled together via a wired and/or wireless bus structure) or may be distributedly located (e.g., cloud computing via indirect coupling via a local area network and/or a wide area network). Further note that if the processing module, module, processing circuit, and/or processing unit implements one or more of its functions via a state machine, analog circuitry, digital circuitry, and/or logic circuitry, the memory and/or memory element storing the corresponding operational instructions may be embedded within, or external to, the circuitry comprising the state machine, analog circuitry, digital circuitry, and/or logic circuitry. Still further note that, the memory element may store, and the processing module, module, processing circuit, and/or processing unit executes, hard coded and/or operational instructions corresponding to at least some of the steps and/or functions illustrated in one or more of the Figures. Such a memory device or memory element can be included in an article of manufacture.

One or more embodiments have been described above with the aid of method steps illustrating the performance of specified functions and relationships thereof. The boundaries and sequence of these functional building blocks and method steps have been arbitrarily defined herein for convenience of description. Alternate boundaries and sequences can be defined so long as the specified functions and relationships are appropriately performed. Any such alternate boundaries or sequences are thus within the scope and spirit of the claims. Further, the boundaries of these functional building blocks have been arbitrarily defined for convenience of description. Alternate boundaries could be defined as long as the certain significant functions are appropriately performed. Similarly, flow diagram blocks may also have been arbitrarily defined herein to illustrate certain significant functionality.

To the extent used, the flow diagram block boundaries and sequence could have been defined otherwise and still perform the certain significant functionality. Such alternate definitions of both functional building blocks and flow diagram blocks and sequences are thus within the scope and spirit of the claims. One of average skill in the art will also recognize that the functional building blocks, and other illustrative blocks, modules and components herein, can be implemented as illustrated or by discrete components, application specific integrated circuits, processors executing appropriate software and the like or any combination thereof.

In addition, a flow diagram may include a "start" and/or "continue" indication. The "start" and "continue" indications reflect that the steps presented can optionally be incorporated in or otherwise used in conjunction with other routines. In this context, "start" indicates the beginning of the first step presented and may be preceded by other activities not specifically shown. Further, the "continue" indication reflects that the steps presented may be performed multiple times and/or may be succeeded by other activities not specifically shown. Further, while a flow diagram indicates a particular ordering of steps, other orderings are likewise possible provided that the principles of causality are maintained.

The one or more embodiments are used herein to illustrate one or more aspects, one or more features, one or more concepts, and/or one or more examples. A physical embodi-

11

ment of an apparatus, an article of manufacture, a machine, and/or of a process may include one or more of the aspects, features, concepts, examples, etc. described with reference to one or more of the embodiments discussed herein. Further, from figure to figure, the embodiments may incorporate the same or similarly named functions, steps, modules, etc. that may use the same or different reference numbers and, as such, the functions, steps, modules, etc. may be the same or similar functions, steps, modules, etc. or different ones.

Unless specifically stated to the contra, signals to, from, and/or between elements in a figure of any of the figures presented herein may be analog or digital, continuous time or discrete time, and single-ended or differential. For instance, if a signal path is shown as a single-ended path, it also represents a differential signal path. Similarly, if a signal path is shown as a differential path, it also represents a single-ended signal path. While one or more particular architectures are described herein, other architectures can likewise be implemented that use one or more data buses not expressly shown, direct connectivity between elements, and/or indirect coupling between other elements as recognized by one of average skill in the art.

The term “module” is used in the description of one or more of the embodiments. A module implements one or more functions via a device such as a processor or other processing device or other hardware that may include or operate in association with a memory that stores operational instructions. A module may operate independently and/or in conjunction with software and/or firmware. As also used herein, a module may contain one or more sub-modules, each of which may be one or more modules.

As may further be used herein, a computer readable memory includes one or more memory elements. A memory element may be a separate memory device, multiple memory devices, or a set of memory locations within a memory device. Such a memory device may be a read-only memory, random access memory, volatile memory, non-volatile memory, static memory, dynamic memory, flash memory, cache memory, and/or any device that stores digital information. The memory device may be in a form a solid state memory, a hard drive memory, cloud memory, thumb drive, server memory, computing device memory, and/or other physical medium for storing digital information.

While particular combinations of various functions and features of the one or more embodiments have been expressly described herein, other combinations of these features and functions are likewise possible. The present disclosure is not limited by the particular examples disclosed herein and expressly incorporates these other combinations.

What is claimed is:

1. A non-transitory computer readable storage device comprises:

a first memory section that stores operational instructions that, when executed by a computing device of a dispersed storage network (DSN), causes the computing device to:

as data accesses occur for a plurality of data objects of a storage container within the DSN;

update object values for at least some of the plurality of data objects to produce updated object values;

update object retention costs for the at least some of the plurality of data objects to produce updated object retention costs;

adjust a dynamic retention threshold based on the updated object values and the updated object retention costs; and

12

update a data object retention policy for a data object of the at least some of the plurality of data objects based on the adjusted dynamic retention threshold to produce an updated retention policy for the data object; and

a second memory section that stores operational instructions that, when executed by the computing device, causes the computing device to:

when one of the data accesses is a deletion request regarding the data object, utilize the updated retention policy to determine a deletion-retention option for the data object; and

execute the deletion-retention option on the data object.

2. The non-transitory computer readable storage device of claim 1, wherein when the one of the data accesses is the deletion request regarding the data object, the second memory section stores further operational instructions that, when executed by the computing device, cause the computing device to:

determine whether an object value of the updated object values for the data object and an object retention cost of the updated object retention costs for the data object compares favorably to the dynamic retention threshold, wherein the comparing favorably indicates the object value and the object retention costs exceed the dynamic retention threshold; and

when the object value and the object retention cost of the data object compares favorably to the dynamic retention threshold, discard the deletion request.

3. The non-transitory computer readable storage device of claim 2, wherein the second memory section stores further operational instructions that, when executed by the computing device, cause the computing device to:

when the object value and the object retention cost of the data object compares unfavorably to the dynamic retention threshold, utilize a current updated retention policy for the data object to determine the deletion-retention option for the data object, wherein the comparing unfavorably indicates the object value and the object retention cost does not exceed the dynamic retention threshold.

4. The non-transitory computer readable storage device of claim 1, wherein the first memory section stores further operational instructions that, when executed by the computing device, cause the computing device to update an object value of the object values regarding a first data object of the plurality of data objects by:

determining an updated value factor based on one or more of:

frequency of the data accesses for the first data object;

a type of the first data object;

a cost associated with one or more of the data accesses for the first data object;

a regulation regarding deletion-retention of the first data object; and

overwriting occurrences of the first data object; and changing a then current object value based on the updated value factor to produce an updated first object value of the updated object values.

5. The non-transitory computer readable storage device of claim 1, wherein the first memory section stores further operational instructions that, when executed by the computing device, cause the computing device to update an object retention cost of the object retention costs regarding a first data object of the plurality of data objects by:

determining an updated cost factor based on one or more of:

13

cost per access of the first data object; cost per read of the first data object; cost per input-output processing (TOP) of the first data object; storage costs of the first data object; and user subscription criteria regarding the first data object; and

changing a then current object retention cost based on the updated cost factor to produce an updated first object retention costs of the updated object retention costs.

6. The non-transitory computer readable storage device of claim 1, wherein the first memory section stores further operational instructions that, when executed by the computing device, cause the computing device to update the data object retention policy for the data object by:

executing a heuristic function using an updated object value of the updated object values and an updated object retention cost of the updated object retention costs as inputs; and

updating a then current data object retention policy based on a result of the heuristic function to produce the updated retention policy for the data object.

7. The non-transitory computer readable storage device of claim 1 further comprises:

a third memory section that stores operational instructions that, when executed by the computing device, cause the computing device to:

when the one of the data accesses is not the deletion request regarding the data object:

determine whether an object value of the object values and an object retention cost of the object retention costs for the data object compares favorably to the dynamic retention threshold;

when the object value and the object retention of the data object compares favorably to the dynamic retention threshold, maintain the data object; and

when the object value and the object retention cost of the data object compares unfavorably to the dynamic retention threshold:

utilize the updated retention policy for the data object to determine the deletion-retention option for the data object; and

execute the deletion-retention option on the data object.

14

8. The non-transitory computer readable storage device of claim 1 further comprises:

a third memory section that stores operational instructions that, when executed by the computing device, cause the computing device to:

determine a desired level of profitability for the storage container; and

further adjust the dynamic retention threshold such that an amount of data objects of the plurality of data objects of the storage container having a corresponding object value and a corresponding object retention cost that compares favorably to the dynamic retention threshold produces the desired level of profitability.

9. The non-transitory computer readable storage device of claim 1 further comprises:

a third memory section that stores operational instructions that, when executed by the computing device, cause the computing device to:

update, for at least some of the data accesses, a storage container value to produce an updated storage container value;

update, for at least some of the data accesses, a storage container cost to produce an updated storage container cost; and

update, for at least some of the data accesses, the data object retention policy for the data object based on an updated object value, an updated object retention cost, the updated storage container value, and the updated storage container cost.

10. The non-transitory computer readable storage device of claim 1, wherein the updated retention policy for the data object further comprises:

a deletion with archiving option;

a deletion without archiving option;

a deletion and data object scrub without archiving option;

a deletion prohibition option;

a deletion restriction option based on one or more user parameters; and

a subsequent data disposal option.

* * * * *